Jason Walker

Jason.walker@blackpool.ac.uk

App structure

Warehouse Application

Contents

[BaseInventoryItem.py 2](#_Toc182578658)

[Step 1: Define the Class 2](#_Toc182578659)

[Step 2: Add Placeholder Methods 2](#_Toc182578660)

[Step 3: Define the String Representation 2](#_Toc182578661)

[Key Points to Remember 3](#_Toc182578662)

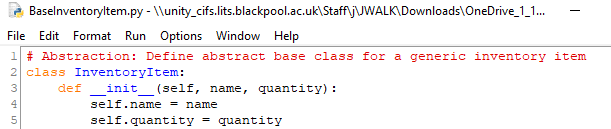
[Integration 3](#_Toc182578663)

# BaseInventoryItem.py

This document provides an in-depth explanation of the BaseInventoryItem script, breaking down each step of the implementation for clarity.

# Step 1: Define the Class

The script begins by defining a base class named 'InventoryItem'. This class serves as an abstract representation of inventory items. It provides a template for other classes that will inherit from it.



Explanation:

1. `class InventoryItem:` - This line declares the class named 'InventoryItem'.

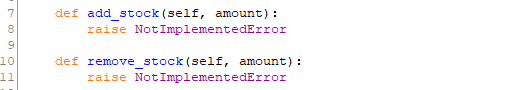
2. `def \_\_init\_\_(self, name, quantity):` - This is the constructor method that initializes two attributes:

- `name`: Represents the name of the inventory item.

- `quantity`: Represents the number of items in stock.

# Step 2: Add Placeholder Methods

To ensure that subclasses implement specific behaviors, the base class includes methods that raise `NotImplementedError`. These methods act as placeholders.



Explanation:

1. `add\_stock`: This method is intended to add stock to an inventory item. Subclasses will provide specific implementations.

2. `remove\_stock`: This method is intended to remove stock from an inventory item. Subclasses will handle validation and specific logic.

# Step 3: Define the String Representation

The `\_\_str\_\_` method is defined to provide a readable representation of an inventory item. This is particularly useful for debugging or displaying inventory details.



Explanation:

1. `def \_\_str\_\_(self):` - This method is automatically called when the object is converted to a string.

2. `return f'{self.name}: {self.quantity}'` - Returns a string in the format 'ItemName: Quantity', where `name` and `quantity` are attributes of the object.

# Key Points to Remember

1. The `InventoryItem` class is abstract and cannot be used directly. Subclasses must implement the `add\_stock` and `remove\_stock` methods.

2. The class is designed to be extended, ensuring that specific types of inventory items (e.g., RegularItem, PerishableItem) can inherit and implement their own logic.

3. The `\_\_str\_\_` method enhances the usability of the class by providing a clear representation of the object.

# Integration

This class will be used as a base for creating specific inventory item classes, such as 'RegularItem' or 'PerishableItem'. These subclasses will inherit its structure and implement the required methods.